



Build Safely With AI

Leveraging Chainguard and Cursor for a safer SDLC with agentic coding

About Me

- Staff DevRel Engineer at Chainguard, joined 4 years ago
- Background in systems administration, PHP, devOps
- Building side projects and internal tools with AI
- 3D designer / printer enthusiast



Table of Contents

- How the threat model changes with AI assisted coding
- From threat to principle: closing the trust gap
- A trusted AI-assisted SDLC
- Demo: Building an SBOM visualizer



"This Changes Everything"

Your AI assistant has the same privileges you do, and a big appetite for running things

The threat model just changed

- AI assistants don't just suggest code anymore. They install packages, run scripts, edit configs, and execute commands
- Review fatigue is real; the volume of code written by AI makes it hard for human eyes to keep track of everything
- Your laptop holds SSH keys, cloud credentials, browser cookies, source code, customer data
- Every install and tool call (and every fetched URL) is now an attack surface with **your privileges**

The threat model just changed

"...as agents become capable of doing work that once required a person or even a team, the cost of not deploying grows large enough that the risk-reward calculation tips heavily toward adoption, as long as products can be made safe. The engineering question becomes **how to cap the blast radius.**"




How we contain Claude across products, [Anthropic Blog](#)

Your local dev environment at risk

- **npm pre/post-install scripts** — npm install runs arbitrary code before you've read a single line. Recent examples: nx, ctrl/tinycolor, Shai-Hulud, etc.
- **Dead man's switch packages** — malicious payload triggers on a condition (date, missing heartbeat, removed maintainer). You can't audit your way out of code that hasn't run yet.

Developing directly on your host machine can open the doors to malicious script execution in your environment's context, with access to credentials and other sensitive data.

AI-native attack vectors

-  **Slopsquatting**
 - LLM hallucinates a package name → attacker registers it → you install malware.
-  **Prompt injection**
 - Untrusted content the agent reads becomes instructions the agent runs.
-  **Config hijacking**
 - Malicious package rewrites your agent's config — every future session is compromised.



From threat to principle: closing the trust gap

Principles for safer AI-assisted coding

The Pattern

- npm scripts run because npm trusts the package
- Slopssquatting works because the AI trusts its own output
- Prompt injection works because the agent trusts what it reads
- Config hijacking works because tools trust their own config

Every vector exploits unverified trust — somewhere in the chain, something assumed instead of checked.

Two principles for a safe SDLC with AI assistance



ISOLATION

minimizing blast radius

- Each project in its own isolated dev env
- Minimal + Reproducible + Disposable
- Keeping same prod principles to minimize risk of container escape



PROVENANCE

pulling from trusted sources

- Minimal, signed container images with provenance attestations
- Packages built from source in tamper-proof environment (avoid build-time vulns)
- Verifiable app-level dependencies with provenance attestations

A trusted AI-assisted SDLC

Vibecoding with trusted sources

Vibecoding with trusted sources

- The agent picks the base image → it's a Chainguard Container
- The agent pulls a library → it's a Chainguard Library, signed and verifiable
- The agent executes app commands in a container → not on your laptop

Maintain speed, change trust model.

Vibecoding with trusted sources

Chainguard

cgr.dev/chainguard-private/python:latest

Latest CVE count	Daily average	Compressed size
4	5	23.91 MB

Alternative

python:latest

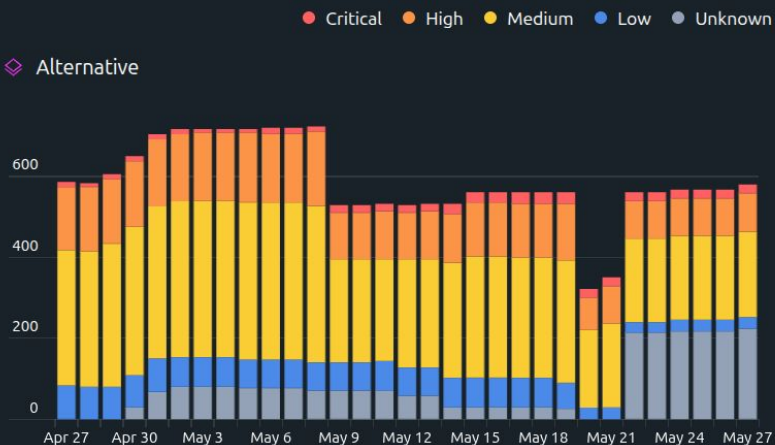
Latest CVE count	Daily average	Compressed size
579	588	395.18 MB

CVEs by severity

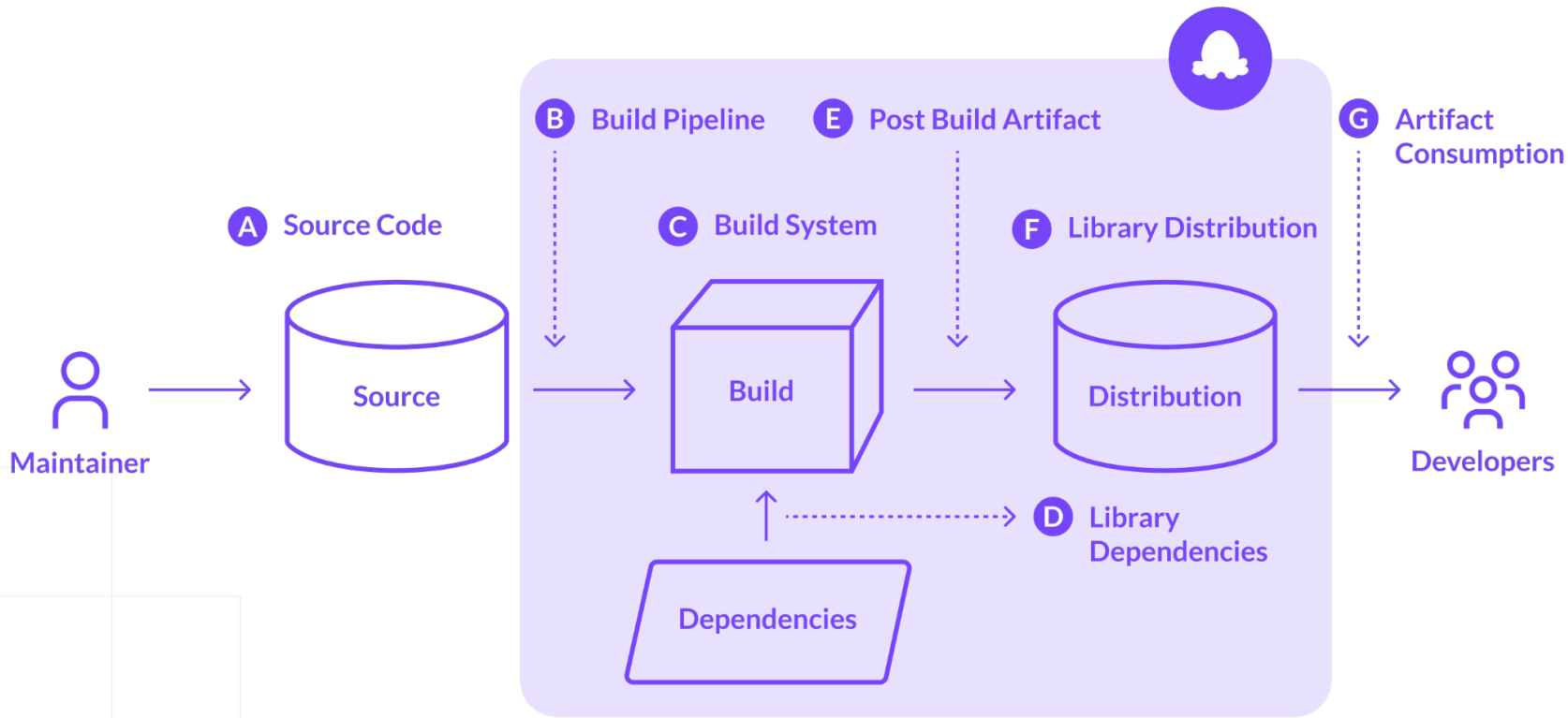
Chainguard



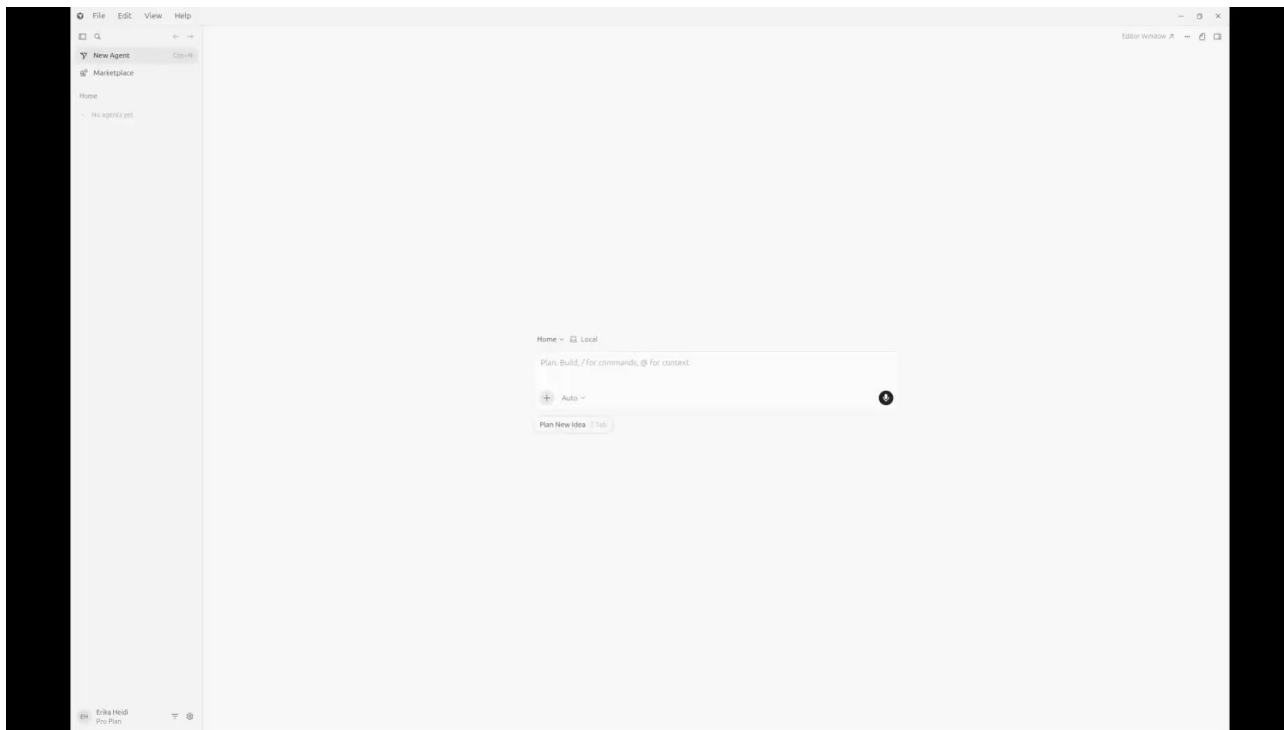
Alternative



Vibecoding with trusted sources



The Chainguard plugin for Cursor



9 Skills and 4 MCP servers

Skills

- chainguard-check-advisories
- chainguard-compare-versions
- chainguard-find-image
- chainguard-inspect-image
- chainguard-libraries-java
- chainguard-libraries-javascript
- chainguard-libraries-python
- chainguard-migrate-dockerfile
- chainguard-setup-auth

MCPs

- **Cg Api:** Authenticated access to your Chainguard organization – security advisories, CVE status, policies, IAM, and org management.
- **Cg Apk:** Searches Chainguard registry for package names, versions, dependencies, and Debian/Alpine equivalents.
- **Cg Oci:** Queries the live cgr.dev registry for image availability, tags, digests, architectures, and SBOM.
- **Cg Versions:** Retrieves version history, package-level changelogs, and digest history for Chainguard images



Demo: Building an SBOM Visualizer


Using Chainguard Containers and Chainguard Libraries for a trusted AI-assisted Workflow

Take Off Checklist

- **Have a Chainguard Account**
 - chainctl installed and ready to use
- **Have a Cursor account**
 - IDE installed and ready to go
- **Install Chainguard plugin**
 - Connect all MCP servers <- this is important
- **Optional to speed things up**
 - chainctl auth login
 - chainctl config set default.group <org>

The Initial Prompt

Create a Python web application called "sbomviz" that ingests an spdx SBOM file via upload form and shows a human-friendly report of everything, with a search box that lets you search for a string in the dependencies. Use the files in the "samples" folder for testing. The app must run on a containerized development environment using Chainguard Containers. You are not allowed to run the application or install dependencies on the host. When finished, create a README and an AGENTS file with instructions to enforce this rule.

File Edit View Help Python web application for SBO... 

← →

Create a Python web application called "sbomviz" that ingests an spdx SBOM file via upload form and shows a human-friendly report of everything, with a search box that lets you search for a string in the dependencies. Use the files in the "samples" folder for testing. The app must run on a containerized development environment.

- Added Python dependency file:
 - `requirements.txt` (`Flask==3.1.1`)
- Added containerized dev environment using Chainguard:
 - `Dockerfile` based on `cgr.dev/chainguard/python:latest-dev`
 - `docker-compose.yml` to build/run on port `8000`
 - `.dockerignore`
- Added docs/rules:
 - `README.md` with usage and explicit container-only policy
 - `AGENTS.md` enforcing:
 - no host dependency installs
 - no host app execution
 - container-only execution commands


Notes on your constraint



- I did **not** run the application or install dependencies on host.
- I also did not execute container commands here; setup is prepared for you to run via Docker Compose.

How to run

From the repo root:

- `docker compose up --build`
- Open `http://localhost:8000`
- Upload `samples/python.spdx` or `samples/static.spdx`
- Use search values like `glibc`, `CONTAINS`, or `SPDXRef-Package-apk` in the dependency search box.

+ Send follow-up Auto 

 Local  16%

EH Erika Heidi Pro Plan

Repositories

- sbomviz
- Python web application f...

Home

No agents yet

The Vibecoded App

sbomviz report

[Upload another file](#)

Document

Name sbom-sha256:6dd48efa2b7db82be32a817b91c1a37d24c2ed188847dd3422f9209d39968db9
SPDX Version SPDX-2.3
Created 2026-05-28T07:37:08Z
Creators Tool: apk (v1.2.9), Organization: Chainguard, Inc
Total Packages 128
Total Relationships 139

Dependency Search

Find text in dependency relationships

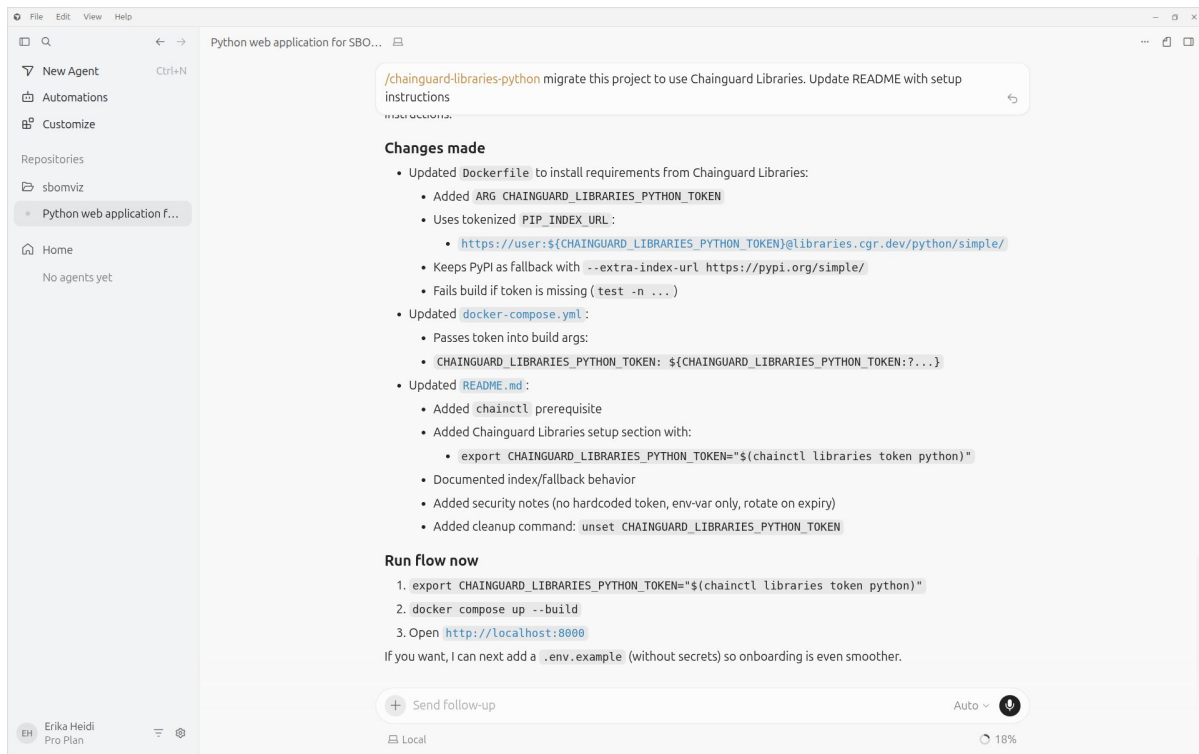
Showing 12 of 139 relationship(s).

	Relationship	To
	DESCRIBED_BY	glibc.yaml@177abde51d27cedf625bcf2e7f21fdb2fa26243d
	GENERATED_FROM	glibc@4070d808bea1c077eb7e7d52b52b91cae98205d5
	CONTAINS	glibc-locale-posix@2.43-r8
	CONTAINS	glibc-locale-posix@2.43-r8
a1a8d37cb3675efcaddae60ab46dd195ee2550334f029e60	CONTAINS	glibc-locale-posix@2.43-r8
	DESCRIBED_BY	glibc.yaml@177abde51d27cedf625bcf2e7f21fdb2fa26243d
	GENERATED_FROM	glibc@4070d808bea1c077eb7e7d52b52b91cae98205d5
	CONTAINS	glibc@2.43-r8
	CONTAINS	glibc@2.43-r8
a1a8d37cb3675efcaddae60ab46dd195ee2550334f029e60	CONTAINS	glibc@2.43-r8
	DESCRIBED_BY	glibc.yaml@177abde51d27cedf625bcf2e7f21fdb2fa26243d
	GENERATED_FROM	glibc@4070d808bea1c077eb7e7d52b52b91cae98205d5

Migrating to Chainguard Libraries

`/chainguard-libraries-python migrate this project to use Chainguard Libraries. Update README with setup instructions`

Migrating to Chainguard Libraries



The screenshot shows a code editor window titled "Python web application for SBO...". A notification bubble at the top reads: "/chainguard-libraries-python migrate this project to use Chainguard Libraries. Update README with setup instructions".

Changes made

- Updated `Dockerfile` to install requirements from Chainguard Libraries:
 - Added `ARG CHAINGUARD_LIBRARIES_PYTHON_TOKEN`
 - Uses tokenized `PIP_INDEX_URL`:
 - `https://user:${CHAINGUARD_LIBRARIES_PYTHON_TOKEN}@libraries.cgr.dev/python/simple/`
 - Keeps PyPI as fallback with `--extra-index-url https://pypi.org/simple/`
 - Fails build if token is missing (`test -n ...`)
- Updated `docker-compose.yml`:
 - Passes token into build args:
 - `CHAINGUARD_LIBRARIES_PYTHON_TOKEN: ${CHAINGUARD_LIBRARIES_PYTHON_TOKEN:?..}`
- Updated `README.md`:
 - Added `chainctl` prerequisite
 - Added Chainguard Libraries setup section with:
 - `export CHAINGUARD_LIBRARIES_PYTHON_TOKEN=$(chainctl libraries token python)`
 - Documented index/fallback behavior
 - Added security notes (no hardcoded token, env-var only, rotate on expiry)
 - Added cleanup command: `unset CHAINGUARD_LIBRARIES_PYTHON_TOKEN`

Run flow now

- `export CHAINGUARD_LIBRARIES_PYTHON_TOKEN=$(chainctl libraries token python)`
- `docker compose up --build`
- Open `http://localhost:8000`

If you want, I can next add a `.env.example` (without secrets) so onboarding is even smoother.

At the bottom, there is a "Send follow-up" button, a "Local" indicator, and a "18%" progress indicator.

Migrating to Chainguard Libraries

remove pypy fallback

Migrating to Chainguard Libraries

No agents yet

remove pypy fallback ↩

Editing documentation 1s

Understood — I'll remove the PyPI fallback so installs are Chainguard Libraries only, and update the README to match that strict behavior.

Edited 2 files, explored 2 files +1 -3

Done — removed PyPI fallback entirely.

Updated

- `Dockerfile`
 - Removed `--extra-index-url https://pypi.org/simple/`
 - Installs now resolve only from Chainguard Libraries via authenticated `PIP_INDEX_URL`
- `README.md`
 - Updated setup text to explicitly state:
 - Chainguard Libraries is the **only** index
 - no PyPI fallback

Now your build/coverage will reflect strict Chainguard Libraries availability only.

+ /chainguard Auto ↕

Local 22%

EH Erika Heidi Pro Plan

Migrating to Chainguard Libraries

```
> chainctl libraries verify sbomviz-dev:local
```

```
=====
Python Package Verification Summary
=====
```

```
✓ Verified (77.8% - 7/9 packages):
```

```
📄 Per-Package SBOMs (7 packages):
```

- markupsafe 3.0.3 → Version: 3.0.3 (verified via per-package SBOM - built from source by Chainguard)
- werkzeug 3.1.8 → Version: 3.1.8 (verified via per-package SBOM - built from source by Chainguard)
- blinker 1.9.0 → Version: 1.9.0 (verified via per-package SBOM - built from source by Chainguard)
- click 8.4.1 → Version: 8.4.1 (verified via per-package SBOM - built from source by Chainguard)
- flask 3.1.1 → Version: 3.1.1 (verified via per-package SBOM - built from source by Chainguard)
- itsdangerous 2.2.0 → Version: 2.2.0 (verified via per-package SBOM - built from source by Chainguard)
- jinja2 3.1.6 → Version: 3.1.6 (verified via per-package SBOM - built from source by Chainguard)

```
✗ Unverified (22.2% - 2/9 packages):
```

- setuptools 82.0.1 (not found in SBOM database)
- pip 26.1.1 (not found in SBOM database)

```
=====
Artifact: sbomviz-dev:local
```

```
Verification Coverage: 77.78%
```

 Chainguard

Recap: what we just neutralized

npm install scripts	Chainguard Libraries protects against malicious npm post-install scripts primarily by never building packages that contain install-time scripts, and by building from source
Slopsquatting	every dependency resolved through Chainguard Libraries, verified by chainctl libraries verify
Prompt injection and Config Hijacking	blast radius bounded by the container; no SSH keys, no AWS creds to exfiltrate

Other Skills to Try

- **chainguard-find-image**
 - Finds a suitable image; reference by digest
- **chainguard-inspect-image**
 - Inspect SBOM, packages and configuration for a CGR Image
- **chainguard-check-advisories**
 - Checks CVEs and security advisories for a Chainguard image or APK package
- **chainguard-migrate-dockerfile**
 - Migrate existing Dockerfiles to use Chainguard Containers
- **chainguard-compare-versions**
 - Used to compare two versions of the same image, see what changed



Wrapping UP

Upcoming events

JUNE 3 @ 1PM ET

Securing Your Software Supply Chain for CMMC 2.0

With CMMC Phase 2 certification requirements taking effect Nov. 10, Chainguard and Schellman break down how to meet FIPS-validated cryptography and zero-CVE requirements.



JUNE 25 @ 1PM ET

Tech Talks: When AI builds faster than you can secure it

Join us for:

- Keynote on proactive AI software supply chain security and Chainguard's new capabilities
- Three conversations security teams need to have about Mythos (with KKR)
- What engineering leaders must get right before scaling AI development (with Cursor)



FREE OFFER

Chainguard Libraries are available for free until June 30, 2026

Access Python, Java, and JavaScript libraries
rebuilt from verified source and safe from the
next supply chain attack



FREE OFFER

Chainguard Catalog Starter: Your choice of 5 free container images

Instant access to trusted container images
from the industry's most comprehensive
catalog





Thank you!

Erika Heidi | erika@chainguard.dev